

PROGRAMAREA CALCULATOARELOR

Tema 3 - Editor de imagini (Fișiere, Parsare de stringuri și Structuri)

Termen de predare:

20 Decembrie 2024, ora 23:59 (**deadline soft**)

8 Ianuarie 2025, ora 23:59 (**deadline hard**)

Responsabili Temă:

Alex Deonise, Alexandra Ion, Călin Precupețu,
Maria Teodor, Bălteanu Vlad, Martinuț Francesco, Radu-Andrei Georgescu

Changelog:

- 09.12.2024: Publicare tema 3.

Vă rugăm să adresați toate întrebările legate de teme pe forumul asociat temei 3 de pe site-ul de curs. Orice clarificare / corectare / actualizare legată de enunț, checker sau lucruri administrative etc, vor fi anunțate exclusiv pe forumul dedicat temei 3.

Vă rugăm să citiți cu mare atenție secțiunea de regulament și fișierul README-checker.md!

Cuprins

Obiectivele temei de casă	2
Descrierea problemei	2
Funcționarea programului	2
Formatele de imagini	5
Utilizarea nucleelor (comanda APPLY)	7
Histograma și egalizarea imaginilor	9
Generarea propriilor imagini de test	14
Sumarul mesajelor și exemplu de rulare	15
Restricții și precizări	16
Regulament	17

Obiectivele temei de casă

În urma realizării temei de casă, studenții:

- Vor putea descrie și vor ști să prelucreze (folosind limbajul C) imagini în formatul NetPBM;
- Vor fi capabili să realizeze parsarea unui set de comenzi specifice unui program;
- Vor putea scrie programe ce prelucrează fișiere (text sau binare), prin respectarea unei specificații;
- Vor fi exersat utilizarea tipurilor de date definite de utilizator (structuri, uniuni și enumerații) în limbajul C;
- Vor fi implementat un program cu interfață text pentru realizarea unor transformări de bază pe fișiere de tip imagine.

Descrierea problemei

Gigel își dorește să realizeze primul lui proiect de anvergură, care să atragă ochiul tuturor recruterilor de peste mări și țări. Pentru asta, s-a gândit să reinventeze TikTok-ul. Într-o primă etapă, doar cu imagini.

El s-a inspirat din modul de funcționare al Adobe Photoshop, din care a reținut 2 pași importanți pentru fiecare dintre transformări:

1. Selectarea regiunii de lucru;
2. Executarea transformării.

Gigel este abia anul 1 la ACS, așa că nu va implementa interfața grafică a aplicației. În schimb, și-a ales câteva transformări simple pe care să le aplice asupra imaginilor RGB sau Grayscale.

Căutând informații despre formatele de imagine, Gigel a descoperit PGM și PPM. Acestea sunt 2 formate foarte similare pentru stocarea de imagini, fiecare având câte o versiune plain (în care valorile pixelilor sunt stocate în format ASCII) și o versiune binară (în care valorile sunt stocate în mod compact).

Documentațiile ce descriu formatele PPM și PGM pot fi găsite la următoarele adrese:

1. <http://netpbm.sourceforge.net/doc/pgm.html>
2. <http://netpbm.sourceforge.net/doc/ppm.html>

În cadrul temei se propune implementarea unui editor de text, care să suporte efectuarea unui set de operații peste imagini formate PPM sau PGM, având valoarea maximă a intensității unei culori cel mult 255.

Fișierele de intrare și de ieșire ale programului vor respecta standardul PBM, fiind identificate prin magic-word-urile P2, P3, P5, P6. Nu trebuie să implementați suport pentru alte formate (ex. JPEG, PNG etc.).

Funcționarea programului

Observație: În această secțiune, atunci când vedeți un text de forma *<valoare>* înseamnă că acolo va fi prezentă în mod obligatoriu o valoare ce trebuie înlocuită. În mod similar, notăm cu *[valoare]* o valoare ce trebuie înlocuită, dar poate lipsi. Caracterele ce marchează efectiv înlocuirea nu trebuie să apară în comenzile date sau în output-ul programului.

Programul va porni fără parametri și va suporta introducerea, câte una pe linie, a unor comenzi. Pentru fiecare comandă, acesta va face operația cerută după care va afișa pe ecran un mesaj (de succes sau de eroare). Mesajele ce vor fi afișate sunt specificate în descrierea comenzilor.

Comenzile suportate vor fi:

- **LOAD <fișier>**
 - Va încărca în memoria programului imaginea (PPM sau PGM) din fișierul dat.
 - Dacă deja există un fișier încărcat în memorie, atunci acesta va fi înlocuit.
 - Output-ul comenzii va fi:
 - * Loaded <fișier> dacă operația s-a executat cu succes.
 - * Failed to load <fișier> dacă operația a întâmpinat dificultăți.
 - La încărcarea unui fișier nou, vom considera că acesta este automat selectat complet (întreaga suprafață);
 - În cazul în care operația eșuează, programul va reveni la starea în care nu are niciun fișier încărcat.
- **SELECT <x1> <y1> <x2> <y2>**
 - Restrânge efectul comenzilor următoare la pixelii incluși în intervalul [x1, x2) pe axa X (lățimea imaginii), respectiv în intervalul [y1, y2) pe axa Y (înălțimea imaginii).
 - * Prin SELECT <x1> <y1> <x2> <y2>, se vor selecta pixelii [x1, x2), [y1, y2). Spre exemplu, SELECT 0 0 1 1 va selecta doar pixelul (0, 0); în vreme ce SELECT 0 0 2 2 va selecta pixelii (0, 0), (0, 1), (1, 0), (1, 1).
 - Originea axelor de coordonate se află în colțul din stânga sus.
 - Output-ul comenzii va fi:
 - * Selected <x1> <y1> <x2> <y2> dacă operația s-a executat cu succes.
 - * Invalid set of coordinates dacă vreunul din punctele de delimitare se află în afara imaginii.
 - * No image loaded dacă nu există un fișier încărcat.
 - După o selecție invalidă, se va păstra selecția anterioară.
 - Ordinea (x1 < x2 și y1 < y2) nu este garantată. Va trebui să tratați ambele cazuri.
- **SELECT ALL**
 - Readuce zona de selecție la întreaga imagine.
 - Output-ul comenzii va fi:
 - * Selected ALL dacă operația s-a executat cu succes.
 - * No image loaded dacă nu există un fișier încărcat.
- **HISTOGRAM <x> <y>**
 - Va afișa histograma imaginii folosind maxim x stelute și y bin-uri;
 - Valorile permise pentru y sunt puterile lui 2 din intervalul [2, 256];
 - Output-ul comenzii va fi:
 - * Histograma, dacă aceasta se poate calcula;
 - * Black and white image needed dacă imaginea este color;
 - * No image loaded dacă nu există un fișier încărcat.
- **EQUALIZE**
 - Va face egalizarea imaginii alb-negru conform metodei descrise în **secțiunea aferentă a temei**;
 - Output-ul comenzii va fi:
 - * Equalize done, dacă operația s-a efectuat cu succes;
 - * Black and white image needed dacă imaginea este color;
 - * No image loaded dacă nu există un fișier încărcat.
 - Operația se va realiza mereu pe toată imaginea, indiferent de selecția curentă. Deși operația se va aplica asupra întregii imagini, nu se va realiza niciun fel de SELECT (și, în particular, niciun SELECT ALL), astfel că selecția curentă a imaginii nu va fi modificată.
- **ROTATE <unghi>**
 - Va face rotirea selecției curente cu un anumit număr (exprimat în grade).
 - Valorile permise pentru unghi sunt +/-90, +/-180, +/-270, +/-360.

- În cazul în care valoarea unghiului este pozitivă, rotirea selecției se va face spre dreapta. Dacă valoarea unghiului este negativă, rotirea se va face spre stânga.
- Output-ul comenzii va fi:
 - * Rotated <unghi>, dacă operația s-a efectuat cu succes;
 - * The selection must be square, dacă selecția nu este compatibilă cu cerința;
 - * Unsupported rotation angle, dacă valoarea unghiului de rotație nu este unul dintre cele permise.
 - * No image loaded dacă nu există vreun fișier încărcat.
- **Observație:** ROTATE se poate aplica doar pentru o selecție de dimensiune NxN (submatrice pătratică) sau pe întreaga imagine.
- CROP
 - Va reduce imaginea la cea cuprinsă în cadrul selecției curente.
 - După CROP, orice comandă, inclusiv SELECT ALL, va face referire doar la subimaginea rezultată în urma comenzii CROP.
 - Când operația se execută cu succes, comanda va avea drept output mesajul: Image cropped.
 - Dacă înainte de execuția comenzii nu se încarca nicio imagine, output-ul comenzii va fi No image loaded.
- APPLY <PARAMETRU>
 - Disponibilă doar pentru tipurile de imagini color. Această comandă va aplica peste selecție unul din nucleele de imagine prezentate în secțiunea "Formatele de imagini", în funcție de parametrul introdus:
 - * <EDGE>;
 - * <SHARPEN>;
 - * <BLUR>;
 - * <GAUSSIAN_BLUR>;
 - Modul de aplicare a transformării este următorul:
 - * Se înmulțesc valorile pixelilor din imagine cu valorile corespunzătoare a pixelilor din kernel
 - * Mai apoi, rezultatul noului pixel este suma elementelor menționate
 - Matricea kernel se va aplica pe toate cele 3 canale ale imaginii color.
 - Output-ul comenzii va fi "APPLY <PARAMETRU> done";
 - În cazul în care nu se introduce parametrul corespunzător se va afișa "APPLY parameter invalid".
 - În cazul în care APPLY se aplică asupra imaginilor grayscale, se va afișa "Easy, Charlie Chaplin".
 - Dacă înainte de execuția comenzii nu se încarca nicio imagine, output-ul comenzii va fi No image loaded.
 - Puteți găsi o explicație detaliată despre cum funcționează aceste nuclee [aici](#);
 - Pixelii care nu au destui vecini pentru a putea calcula noile valori, vor rămâne nemodificați;
 - **Observație:** Prin aplicarea fitrului EDGE se pot obține valori în afara intervalului [0, 255]. Pentru tratarea acestui caz, se va aplica [funcția clamp](#). Astfel, valorile negative vor deveni 0, iar cele ce depășesc 255 vor fi limitate la această valoare.
- SAVE <nume_fișier> [ascii]
 - Salvează imaginea curentă în fișierul nume_fișier
 - Dacă parametrul ascii este prezent, atunci va salva valorile pixelilor în format ASCII. Altfel, se vor salva în format binar.
 - * Formatul (binary/text) în care se va salva imaginea este specificat la apelul comenzii SAVE. Acest format trebuie respectat ca atare, indiferent de formatul pe care l-a avut imaginea încărcată inițial.
 - În urma operației SAVE, ultima imagine încărcată prin operația LOAD va rămâne încărcată în memorie.
 - Dacă încă nu s-a încărcat o imagine, atunci se va afișa mesajul "No image loaded."
 - Comanda va afișa, după salvarea fișierului mesajul Saved <nume_fișier>.
- EXIT

- Face închiderea grafului a programului.
 - * Acest proces înseamnă dealocarea tuturor resurselor, închiderea fișierelor și oprirea execuției programului.
- Nu se poate realiza fără a avea o imagine încărcată în prealabil.
 - * În acest caz, output-ul comenzii va fi "No image loaded".

O linie care nu poate fi interpretată ca una dintre comenzile de mai sus, va avea ca rezultat afișarea mesajului **Invalid command** și nu va genera alte efecte.

Formatele de imagini

Cele mai multe formate de imagine sunt reprezentate ca puncte (pixeli) definite fie prin intensitatea unei culori, fie printr-o combinație a intensității mai multor culori.

De exemplu:

- O imagine alb negru, poate fi reprezentată prin intensitatea culorii negru, folosind 2 valori (1 pentru negru și 0 pentru alb);
- O imagine în tonuri de gri (eng. grayscale) se poate reprezenta prin intensitatea culorii alb, folosind mai multe valori, pe o scală aleasă (între alb și negru);
- O imagine color se poate reprezenta drept o combinație de Roșu, Verde și Albastru, în anumite cantități (tot alegând niște valori maxime).

Imaginile PPM și PGM fac parte dintr-un standard public, deci cea mai bună sursă de informații pentru a desluși formatele de imagini sunt chiar paginile de referință ale acestor standarde, însă Gigel (care este sursa noastră de inspirație) ne-a rugat să vă lăsăm câteva dintre concluziile lui.

În primul rând, programul vostru trebuie să suporte 4 formate de imagini ce pot fi identificate prin magic word-ul care se află pe prima linie a lor. În Tabelul 1 găsiți caracteristicile formatelor standardizate, așa cum le-a găsit Gigel sumarizate pe Wikipedia.

Tip	Magic Word		Culori
	ASCII	Binary	
Alb-negru	P1	P4	0 & 1 (alb și negru)
Imagine în tonuri de gri (grayscale)	P2	P5	Între 0 și o valoare ce poate varia (noi vom folosi 255), ca o scală între negru-alb
Imagine color	P3	P6	16.777.216 de culori (3 valori între 0 și 255 pentru fiecare dintre cele trei canale Roșu, Verde și Albastru)

Tabela 1: Formate din standardul PBM.

Sursa tabelului: <https://en.wikipedia.org/wiki/Netpbm>

În continuare, vom discuta despre câteva exemple de imagini formate conform standardului.

Listing-ul 1 reprezintă litera L desenată într-o imagine alb-negru, de dimensiune 6 (lățime) x 8 (înălțime).

```

1 P1
2 # This is an example bitmap of a letter
3 6 8
4 0 0 0 0 0 0
5 0 1 0 0 0 0
6 0 1 0 0 0 0

```

```

7 0 1 0 0 0 0
8 0 1 0 0 0 0
9 0 1 0 0 0 0
10 0 1 1 1 1 0 0 0 0 0 0 0

```

Listing 1: Litera L, alb-negru.

Observăm că:

- Prima linie conține obligatoriu un specificator de format (magic-word).
- Fișierele pot conține comentarii (linii care încep cu #), care sunt ignorate.
 - Ca o simplificare pentru temă, vom presupune că toate comentariile se află doar pe linii care au pe prima poziție caracterul "#" și că acestea pot apărea doar pe liniile ce preced matricea de pixeli (nu și în cadrul matricei).
- Pe o linie se află (în format ASCII) cele două dimensiuni ale imaginii (lățime și înălțime)
- Imaginea este reprezentată de width * height valori, reprezentate în ordinea în care acestea apar pe linii, iar trecerea pe un rând nou este opțională (în exemplul de mai sus, ultima linie apare în continuarea penultimei și asta nu face ca imaginea să fie invalidă).

În Listing 2 prezentăm o imagine în tonul de gri.

```

1 P2
2 6 7
3 255
4 0 0 0 0 0 0
5 0 198 85 85 85 0
6 0 198 0 0 0 0
7 0 198 141 141 0 0
8 0 198 0 0 0 0
9 0 198 0 0 0 0
10 0 0 0 0 0 0

```

Listing 2: Litera F, format gray-scale.

Observăm că:

- Valoarea pixelilor corespunde intensității nuanței de gri: 0 va însemna negru, în vreme ce 255 va însemna alb.
- Linia verticală (a literei F) este reprezentată cu tonul de gri cel mai deschis (198/255), prima linie orizontală este cea mai deschisă la culoare (85/255), iar a doua se află între ele ca ton (141/255).

În Listing 3 prezentăm o imagine color (RGB).

```

1 P3
2 3 2
3 255
4 255 0 0 0 255 0 0 0 255
5 255 255 0 255 255 255 0 0 0

```

Listing 3: Imagine color formată din 6 pixeli.

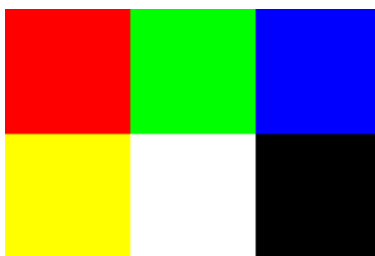


Figura 1: Imaginea aferentă fișierului din Listing 1 (mărită).
Sursa imaginii: <https://en.wikipedia.org/wiki/Netpbm>

Observăm că pentru cei 6 pixeli, sunt specificate 18 valori numerice (câte o valoare pentru fiecare dintre cele 3 canale de culoare - Roșu, Verde și Albastru).

Variantele binare ale acestor formate sunt foarte similare, cu excepția faptului că matricea de pixeli este reprezentată eficient folosind valori binare.

Pentru imaginile noastre, a căror valoare maximă a unui canal este 255, fiecare valoare este reprezentat sub forma unui octet fără semn.

Întrucât nu toate valorile între 0 și 255 fac parte din codul ASCII, aceste fișiere nu pot fi printate ca text. În schimb, putem folosi utilitarul **hexdump** pentru a studia aceste fișiere.

Atunci când apelăm `hexdump -C fișier` acesta afișează:

- Offset-ul din fișier la care se referă linia curentă (pe prima coloană);
- Conținutul fișierului (în octeți reprezentați ca 2 valori hexazecimale) pe coloanele din mijloc;
- Reprezentarea în ASCII a valorilor printabile din fiecare octet (în partea stângă). Pentru valorile `\n` sau cele care nu sunt caractere ASCII, se printează caracterul punct).

```

1 % hexdump -C ascii.ppm
2 00000000  50 33 0a 33 20 32 0a 32  35 35 0a 32 35 35 20 30  |P3.3 2.255.255 0|
3 00000010  20 30 20 30 20 32 35 35  20 30 20 30 20 30 20 32  | 0 0 255 0 0 0 2|
4 00000020  35 35 0a 32 35 35 20 32  35 35 20 30 20 32 35 35  |55.255 255 0 255|
5 00000030  20 32 35 35 20 32 35 35  20 30 20 30 20 30 0a    | 255 255 0 0 0.|
6 0000003f
7
8 % hexdump -C binary.ppm
9 00000000  50 36 0a 33 20 32 0a 32  35 35 0a ff 00 00 00 ff  |P6.3 2.255.....|
10 00000010  00 00 00 ff ff ff 00 ff  ff ff 00 00 00    |.....|
11 0000001d
12
13 % ls -l ascii.ppm binary.ppm
14 -rw-r--r--  1 dorinel  wheel   63 Dec 11 06:11 ascii.ppm
15 -rw-r--r--  1 dorinel  wheel   29 Dec 11 06:06 binary.ppm

```

Listing 4: Hexdump a unor fișiere de tip imagine RGB.

În Listing 4 am inclus hexdump-uri ale imaginii din Figura 1 atât pentru formatul ASCII, cât și pentru cel binar. Observați dimensiunile fișierelor, în special cel al variantei binare.

Utilizarea nucleelor (comanda APPLY)

Un nucleu de imagine (image kernel) este o matrice mică folosită pentru a aplica efecte precum cele pe care le puteți găsi în Photoshop sau Gimp. De asemenea, nucleurile de imagine sunt utilizate în învățarea

automată pentru „extracția caracteristicilor”, o tehnică pentru determinarea celor mai importante porțiuni ale unei imagini. În acest context, procesul este denumit mai general „convoluție” (vezi: rețele neuronale convoluționale).

Pentru a vedea cum funcționează, să începem prin a inspecta imaginea **2** grayscale. Matricea din stânga conține numere, între 0 și 255, care corespund fiecăruia cu luminozitatea unui pixel dintr-o imagine a unei fețe. Imaginea mare, granulată a fost prelucrată (expandată) pentru a fi mai ușor de văzut; ultima imagine are dimensiunea „reală”.



Figura 2: Imagine grayscale pentru înțelegerea nucleelor de imagine.

Sursa imaginii: <https://setosa.io/ev/image-kernels/>

Există o mulțime de nuclee de imagine (vezi <https://setosa.io/ev/image-kernels/>) care pot fi aplicate, însă în această tema ne vom axa pe următoarele tranformări (Figura 3):




Edge detection	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur 3 x 3 (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Figura 3: Nuclee de imagine (Edge, Sharpen, Blur, Gaussian Blur).

Sursa imaginii: [https://en.wikipedia.org/wiki/Kernel_\(image_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))

Aplicarea unui kernel de imagine se efectuează astfel: pentru fiecare bloc de 3x3 pixeli din imaginea, înmulțim fiecare pixel cu intrarea corespunzătoare a nucleului și apoi luăm suma. Această sumă devine un nou pixel în imaginea procesată.

Histograma și egalizarea imaginilor

Calculul unei histograme

Histogramele sunt reprezentări grafice pentru observarea distribuției valorilor dintr-un set de date.

Un caz particular al modului de calcul al histogramelor (și pe care l-ați mai întâlnit) îl reprezintă vectorii de frecvență. De exemplu, considerând următorul set de date: [0, 0, 2, 5, 5, 4, 4, 1, 2, 4, 3, 5], putem obține ușor histograma din Figura 4.

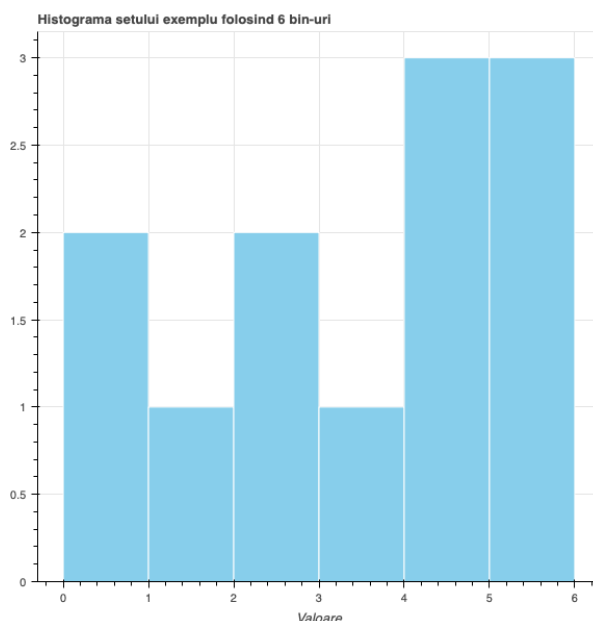


Figura 4: Exemplu de histogramă cu 6 bin-uri.

Rezultatul din imagine provine observarea frecvenței valorilor din set:

- Valoarea 0 apare de 2 ori;
- Valoarea 1 apare o singură dată;
- Valoarea 2 apare de 2 ori;
- Valoarea 3 apare o singură dată;
- Valoarea 4 apare de 3 ori;
- Valoarea 5 apare de 3 ori.

Histograma nu depinde de ordinea valorilor din listă, deci ar fi aceeași pentru orice permutare a ei.

Numărul de intervale utilizate în histogramă (în engleză *bins*) este o alegere importantă și nu este întotdeauna egală cu numărul de valori distincte din set. De exemplu, aceeași histogramă se poate reprezenta folosind 3 intervale ca în Figura 5.

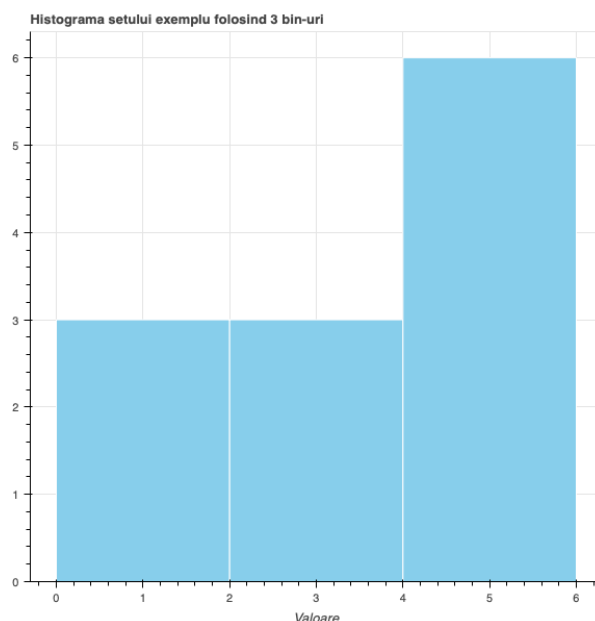


Figura 5: Exemplu de histogramă cu 3 bin-uri.

În cazul în care mai multe valori distincte fac parte dintr-o categorie, atunci histograma însumează numărul de apariții a acestora.

Afișarea unei histograme în Lightroom

Când vine vorba de imagini, histogramele se afișează pentru canalele de culoare ale imaginii sau pentru o altă reprezentare potrivită pentru observator. În Figura 6 puteți observa o histogramă de imagine așa cum este afișată în programul Lightroom.

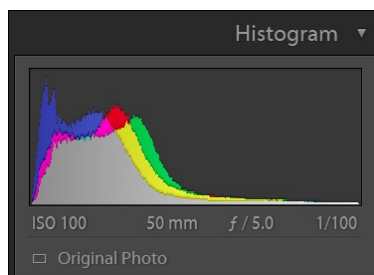


Figura 6: Exemplu de histogramă în Lightroom.

Afișarea histogramei în cadrul temei

Pentru a printa histogramele noastre, vom folosi caracterul ASCII * (asterisk) și un format prestabilit. Astfel, utilizatorul va alege numărul de bin-uri ale histogramei (Y) și numărul maxim de stelute, cel care va fi folosit pentru bin-ul cu cele mai multe elemente (X).

Pe fiecare dintre cele Y rânduri vom afișa:

1. Numărul de stelute de pe acel rând din histogramă;
2. Caracterul tab (`\t`, cod ASCII 9);

3. Caracterul pipe (|, cod ASCII 124);
4. Caracterul tab (\t, cod ASCII 9);
5. Atâtea stelute câte sunt necesare.

Un exemplu de histogramă cu 32 de bin-uri și maxim 32 de stelute poate fi observată mai jos:

```

1 0 |
2 0 |
3 0 |
4 0 |
5 1 | *
6 2 | **
7 5 | *****
8 12 | *****
9 23 | *****
10 28 | *****
11 21 | *****
12 12 | *****
13 11 | *****
14 12 | *****
15 14 | *****
16 13 | *****
17 21 | *****
18 28 | *****
19 23 | *****
20 22 | *****
21 19 | *****
22 18 | *****
23 18 | *****
24 19 | *****
25 15 | *****
26 13 | *****
27 12 | *****
28 11 | *****
29 7 | *****
30 6 | *****
31 5 | *****
32 32 | *****

```

Listing 5: Exemplu de histogramă.

Pentru determinarea numărului de stelute din histograma afișată vom folosi următoarea formulă:

$$numar_{stelute} = \frac{frecventa_{valoare}}{max_{frecventa}} \times x \quad (1)$$

De exemplu, dacă în imagine există 3 valori distincte ale pixelilor, fiecare cu câte 22, 73, respectiv 77 de apariții fiecare, iar utilizatorul dorește afișarea lor cu maxim 7 stelute, atunci vom calcula următoarele valori:

$$\frac{22}{77} \times 7 = 2 \quad \text{stelute}$$

$$\frac{73}{77} \times 7 = 6 \quad \text{stelute}$$

$$\frac{77}{77} \times 7 = 7 \quad \text{stelute}$$

Rezultatul fracțiilor se va rotunji întotdeauna în jos (prin trunchere).

Egalizarea de imagini

În cazul imaginilor obținute ca fotografii cu o expunere incorectă, anumite detalii din ele ar putea fi estompate. Egalizarea tonurilor de imagine reprezintă o metodă de post-procesare, care diminuează efectul expunerii incorecte, redând privitorului accesul la detalii. Un exemplu vizual (înainte–după) al efectului egalizării imaginii poate fi observată în Figura 7.



Figura 7: Exemplu de egalizare (înainte și după). Sursa imaginilor:
https://en.wikipedia.org/wiki/Histogram_equalization

Pe cazul general, egalizarea se poate aplica individual fiecărui canal de culoare sau folosind o anumită funcție de legătură. **În cadrul temei, ne vom ocupa doar de egalizarea imaginilor în tonuri de gri.**

Pentru a ajunge la un astfel de rezultat precum cel din Figura 7, ne vom folosi de histograma valorilor și vom (asupra fiecărui pixel) următoarea transformare:

$$f(a) = 255 \frac{1}{Area} \sum_{i=0}^a H(i) \quad (2)$$

unde:

- $f(a)$ reprezintă noua valoare a pixelului;
- a reprezintă valoarea precedentă a pixelului;
- $Area$ reprezintă suprafața imaginii calculată în pixeli;
- $H(i)$ reprezintă numărul de apariții al valorii i în imagine.

În urma **egalizării histogramei unei imagini**, histograma ei este aplatizată, ca în Figura 8.

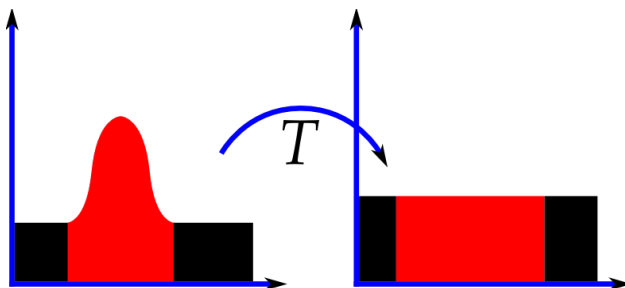


Figura 8: Efectul egalizării asupra histogramei. Sursa figurii:
https://en.wikipedia.org/wiki/Histogram_equalization

Este posibil ca rezultatul $f(i)$ să nu fie o valoare posibilă pe 1 octet. În acest caz vom folosi **funcția clamp** pentru a limita rezultatele în intervalul $[0, 255]$. Calculele se vor face folosind tipul de date **double**, iar rezultatele vor fi rotunjite la scriere, folosind `round()`.

Generarea propriilor imagini de test

Pentru a vă genera propriile fișiere PPM și PGM, recomandăm utilitarul `convert` ce transformă fișierele de tip `png/jpeg/jpg` în formatele preferate de Gigel:

- **Setup:** `sudo apt install imagemagick` (rulat în terminal);
- **Documentație:** `man convert` (idem)
- **Exemple:**
 - `convert foo.jpg foo.ppm` (transformă `foo.jpg` în `foo.ppm`, format binary)
 - `convert foo.jpg foo.pgm` (similar, imagine greyscale)
 - `convert foo.jpg -compress None foo.ppm` (similar, format plain)
 - `convert foo.jpg -compress None foo.pgm`

Sumarul mesajelor și exemplu de rulare

În această secțiune sumarizăm mesajele ce vor fi afișate de către program și dăm un exemplu de rulare, pentru a evidenția modul de funcționare a programului.

Comanda	Output	
	Situație	Mesaj
LOAD <fișier>	Încărcare corectă	Loaded <fișier>
	Operația a întâmpinat dificultăți	Failed to load <fișier>
SELECT <x1><y1><x2><y2>	Selecție corectă	Selected <x1><y1><x2><y2>
	Selecție incorectă	Invalid set of coordinates
	Fișier neîncărcat	No image loaded
SELECT ALL	Selecție realizată	Selected ALL
	Fișier neîncărcat	No image loaded
ROTATE <unghi>	Rotire realizată	Rotated <unghi>
	Selecția nu este pătrată	The selection must be square
	Unghiul ales nu este suportat (nu este divizibil cu 90)	Unsupported rotation angle
	Fișier neîncărcat	No image loaded
EQUALIZE	Egalizare realizată	Equalization done
	Imaginea este color	Black and white image needed
	Fișier neîncărcat	No image loaded
CROP	Operație realizată	Image cropped
	Fișier neîncărcat	No image loaded
APPLY <PARAMETRU>	Operație realizată	APPLY <PARAMETRU> done
	Imagine grayscale	Easy, Charlie Chaplin
	Lipsa parametru	APPLY parameter invalid
	Fișier neîncărcat	No image loaded
SAVE <nume_fișier> [ascii]	Operație realizată	Saved <nume_fișier>
	Fișier neîncărcat	No image loaded
HISTOGRAM <X> <Y>	Operație realizată	Output conform cerinței
	Imagine color	Black and white image needed
	Lipsa parametru sau parametru Y gresit	Invalid set of parameters
	Fișier neîncărcat	No image loaded
SAVE <nume_fișier> [ascii]	Operație realizată	Saved <nume_fișier>
	Fișier neîncărcat	No image loaded
EXIT	Fișier neîncărcat	No image loaded
	Orice altă situație	<nimic>
Comandă necunoscută	Orice situație	Invalid command

Tabela 2: Sumar al mesajelor așteptate de la program.

În Listing 6 prezentăm un exemplu de rulare.

```

1 < LOAD color_test_file
2 > Loaded color_test_file
3 < SELECT 0 1 8 10
4 > Selected 0 1 8 10
5 < CROP
6 > Image cropped
7 < SAVE my_binary_file
8 > Saved my_binary_file
9 < LOAD non_existent_file
10 > Failed to load non_existent_file

```

```
11 < LOAD color_file
12 > Loaded color_file
13 < APPLY SHARPEN
14 > APPLY SHARPEN done
15 < APPLY BLUR
16 > APPLY BLUR done
17 < APPLY GAUSSIAN_BLUR
18 > APPLY GAUSSIAN_BLUR done
19 < SAVE my_plain_file ascii
20 > Saved my_plain_file
21 < LOAD grayscale_test_file
22 > Loaded grayscale_test_file
23 < APPLY EDGE
24 > Easy, Charlie Chaplin
25 < ROTATE 0
26 > Rotated 0
27 < EXIT
```

Listing 6: Exemplu de rulare.

Observație: Am notat cu "<" o comandă scrisă de la tastatură și cu ">" un mesaj afișat la ecran.

Restricții și precizări

- Lucrul cu fișierele este permis strict în cadrul operațiilor LOAD și SAVE. Programul trebuie construit astfel încât să permită lucrul cu imagini mari (hint! alocare dinamică).
- Utilizarea memoriei trebuie să fie eficientă. În cadrul programului se vor reține, la un moment dat, cel mult 2 copii ale imaginii.
- Lucrul corect cu memoria este o cerință obligatorie. Un program care are memory leak-uri sau alte erori detectate de valgrind nu va trece testele de pe vmchecker și nu va primi punctaj.
- Rezolvarea temei trebuie să se facă exclusiv prin codul sursă scris de voi. NU este permisă utilizarea de biblioteci pentru realizarea task-urilor (ex. libnetpbm pentru procesarea de fișiere).
- În implementarea programului **nu** se vor utiliza variabile globale.
- Comenzile SELECT, CROP, ROTATE, APPLY, HISTOGRAM, EQUALIZE SAVE și EXIT vor întoarce „No image loaded.” fără alte efecte laterale dacă sunt introduse înainte de încărcarea unei imagini ce poate fi încărcată.
- Pixelii din afara selecției nu vor fi niciodată afectați de comenzi.
- Toate calculele se vor realiza folosind tipul de date double, iar valorile întregi se vor obține prin rotunjire la cel mai apropiat întreg;
- Fișierele de la intrare pot fi PPM sau PGM atât în format ASCII, cât și în format binary. Identificarea fișierului trebuie să se facă în funcție de antetul său (Atenție! Nu verificați extensia fișierului. Ea poate lipsi.). În testare nu se vor folosi fișiere de input invalide (alte formate în afară de PPM și PGM).
- Checker-ul va verifica rezultatul cu un $\text{EPS} = 4$ (dacă pixelul trebuie să aibă valoarea 20, se acceptă orice valoare între 16 și 24) pentru a întâmpina diferențele provenite din calcule cu virgulă mobilă. Rezultatul programului vostru trebuie să fie identic cu cel de referință în raport cu eroarea permisă.

Regulament

Regulamentul general al temelor se găsește pe ocw (**Temele de casă**). Vă rugăm să îl citiți integral înainte de continua cu regulile specifice acestei teme.

Arhivă

Soluția temei se va trimite ca o arhivă **zip**. Numele arhivei trebuie să fie de forma **Grupă_NumePrenume_TemaX.zip** - exemplu: 311CA_NichitaRadu_Tema3.zip.

Arhiva trebuie să conțină în directorul **RĂDĂCINĂ** doar următoarele:

- Codul sursă al programului vostru (fișierele **.c** și eventual **.h**).
- Un fișier **Makefile** care să conțină regulile **build** și **clean**.
 - Regula **build** va compila codul vostru și va genera următoarele executabile:
* **image_editor**
 - Regula **clean** va șterge **toate** fișierele generate la build (executabile, binare intermediare etc).
- Un fișier **README** care să conțină prezentarea implementării alese de voi. **NU** copiați bucăți din enunț.

Arhiva temei **NU** va conține: fișiere binare, fișiere de intrare/ieșire folosite de checker, checkerul, orice alt fișier care nu este cerut mai sus.

Numele și extensiile fișierelor trimise **NU** trebuie să conțină spații sau majuscule, cu excepția fișierului **README** (care are numele scris cu majuscule și nu are extensie).

Nerespectarea oricărei reguli din secțiunea **Arhivă** aduce un punctaj **NUL** (0) pe temă.

Checker

Pentru corectarea aceste teme vom folosi scriptul **check** din arhiva **PCLP1-check-tema3.zip** din secțiunea de resurse asociată temei. Vă rugăm să citiți **README.md** pentru a ști cum să instalați și utilizați checkerul.

Punctaj

Distribuirea punctajului:

- Punctaj teste: 90p
- Claritatea și calitatea codului: 10p

ATENȚIE! Punctajul maxim pe temă este **100p**. Acesta reprezintă 1.5p din nota finală la această materie. La această temă nu există bonusuri.

Reguli și precizări

- Punctajul pe teste este cel acordat de script-ul **check**, rulat pe **Moodle**. Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare, etc).
- Punctajul pe calitatea explicațiilor și a codului se acordă în mai multe etape:
 - **corectare automată**

- * Checkerul va încerca să detecteze în mod automat probleme legate de coding style și alte aspecte de organizare a codului.
 - * Acesta va puncta cu maxim 10p dacă nu sunt probleme detectate în mod automat.
 - * Punctajul se va acorda proporțional cu numărul de puncte acumulate pe teste din cele 90p.
 - * Checkerul poate să aplice însă și penalizări (exemplu pentru warninguri la compilare) sau alte probleme descoperite la runtime.
- **corectare manuală**
- * Tema va fi corectată manual și se vor verifica și alte aspecte pe care checkerul nu le poate prinde. Recomandăm să parcurgeți cu atenție tutorialul de **coding-style** de pe ocw.cs.pub.ro.
 - * În această etapă se pot aplica depunctări oricât de mari (chiar și totale). Orice rezultat/-punctaj acordat automat de checker, poate fi anulat sau suprascris de către echipa de PCLP la corectarea manuală.
 - * Codul sursă trebuie să fie însoțit de un fișier README care trebuie să conțină informațiile utile pentru înțelegerea funcționalității, modului de implementare și utilizare a programului. Acesta evaluează, de asemenea, abilitatea voastră de a documenta complet și concis programele pe care le produceți și va fi evaluat, în mod analog CS, de către echipa de asistenți. În funcție de calitatea documentației, se vor aplica depunctări sau bonusuri.
 - * La corectarea manuală se va verifica modularizarea. Deprinderea de a scrie cod sursă de calitate, este un obiectiv important al materiei. Sursele greu de înțeles, modularizate neadecvat sau care prezintă hardcodări care pot afecta semnificativ mentenabilitatea programului cerut, pot fi depunctate adițional. Penalizarea pentru modularizare defectuoasă sau absentă complet, poate să fie oricât de mare.
 - **Memory-leaks detectate manual.** *Din rațiuni ce țin de durată a testării, pe vm-checker NU vom avea testare completă cu valgrind (consultați README-checker.md). Temele vor fi testate cu valgrind separat de către asistenți, iar în cazul în care acestea prezintă memory-leaks sau accese invalide de memorie (detectate de valgrind sau prin analiza codului) acestea vor fi depunctate **cu 20% din punctajul total obținut**. Citiți README-checker.md cum puteți activa verificarea cu valgrind pe local.*
 - Lipsa sau conținutul nesatisfăcător al fișierului README (**5p**);
 - O temă care **NU** compilează cu -Wall -Wextra este depunctată la corectarea manuală cu 5p (punctajul echivalent pentru warnings).
 - **Tema trebuie să reprezinte exclusiv munca studentului!** Echipa va aplica regulamentele în vigoare pentru situațiile de fraudă/plagiat, pentru orice nerespectare a acestei reguli, incluzând, dar nelimitându-se la: copierea de la colegi (se aplică pentru ambele persoane implicate) sau din alte surse a unor părți din temă, prezentarea ca rezultat personal a oricărei bucăți de cod provenită din alte surse necitate sau neaprobată în prealabil (site-uri web, alte persoane, cod generat folosit AI/LLM-uri etc.).
 - Orice alt aspect precizat în regulament.
 - Nerespectarea restricțiilor din enunț.
 - Nerespectarea recomandărilor și bunelor practici privind dezvoltarea de programe ce se pot extrage din bibliografie, curs, laborator, regulament sau indicațiile (publice, de pe forum ale responsabililor de temă).

Alte precizări

- Implementarea se va face în limbajul C, iar tema va fi compilată și testată **DOAR** într-un mediu **LINUX**. Nerespectarea acestor reguli aduce un punctaj **NUL**.
- Tema trebuie trimisă sub forma unei arhive pe site-ul cursului curs.upb.ro.
- Tema poate fi submitată de oricâte ori fără depunctări până la deadline. Mai multe detalii se găsesc în regulamentul de pe ocw.
- O temă care **NU** compilează **NU** va fi punctată.
- O temă care compilează, dar care **NU** trece niciun test **NU** va fi punctată.
- Punctajul pe teste este cel acordat de **check** rulat pe **platforma remote** a echipei de **PCLP**.

Echipa de corectare își rezervă dreptul de a depuncta pentru orice încercare de a trece testele fraudulos (de exemplu prin hardcodare).

- Ultima temă submită pe Moodle poate fi rulată de către responsabili/echipa de PCLP de mai multe ori în vederea verificării faptului că nu aveți buguri în sursă. În cazul obținerii punctajelor diferite la rulări multiple, vom păstra minimul dintre punctaje. Vă recomandăm să verificați local tema de mai multe ori pentru a verifica că punctajul este mereu același, apoi să încărcați tema. De asemenea, verificați că punctajul de pe platformă este cel așteptat - singurul punctaj / feedback relevant este cel de pe platforma remote PCLP.